In the 1967 film "The Graduate," Dustin Hoffman's character is offered a single word of advice for a guaranteed future career: plastics. Had the film been set in the early 1990s, the word may well have been "programming." Even long after the dotcom bubble burst, companies such as Google are actively seeking recruits with advanced computer science degrees. The future is bright for programmers -- or so we're told. And yet, some analysts now suggest the picture is not as rosy for recent computer science grads as some would have us believe.

According to the latest data from the U.K.'s Higher Education Statistics Agency (HESA), computer science graduates in the United Kingdom now have the hardest time finding work of graduates in any subject, with an unemployment rate of 17 percent. It should come as no surprise that legal and medical students fare significantly better -- the latter having a jobless rate of practically nil -- but the HESA data suggests that new students might be better off pursuing foreign languages, marketing, or even creative arts, rather than computer science.

While the situation in the United States may not be so dire, in truth few companies share Google's zeal for academic credentials when hiring new developers. Many are willing to accept self-taught programmers, particularly if they have other skills relevant to the business. Some have implemented in-house training programs to allow employees from other disciplines to transition into software development roles. And as development tools themselves become more sophisticated and accessible, even workers with little formal knowledge of programming are trying their hands at creating applications. All are ominous signs that demand for computer science education in the job market may be on the wane.

I should have been an English major

To be sure, an advanced degree in computer science from a prestigious university such as Cal Tech, Carnegie-Mellon, or Stanford is still a valuable asset for any job-seeker. Companies

such as IBM, Google, and Microsoft, which compete on the highest levels of the computing industry, rely on doctoral graduates for the groundbreaking R&D that forms the basis of their cutting-edge products.

Smaller companies with less ambitious goals, on the other hand, may have little need for such specialized expertise, particularly if they aren't in the business of selling software. For such companies, even a four-year degree in computer science may be only a partial qualification. With offshore outsourcing now a mature market, a growing number of businesses see little benefit in retaining entry-level coders at home, preferring to hand off rote coding and algorithm implementation to partners in China, India, Russia, or elsewhere.

Surprisingly, however, even in countries where educated labor is cheap and nuts-and-bolts coding is the norm, a computer science degree is no longer a guaranteed meal ticket. According to Sridhar Vembu, CEO of India-based Web software vendor Zoho, "We noticed that there was little or no correlation between academic performance, as measured by grades and the type of college a person attended, and their real on-the-job performance.
ndian universities have often been criticized for emphasizing memorization over creativity and practical problem-solving, leaving graduates poorly suited for the real-world job market. But Vembu earned his doctorate at Princeton, and some education experts believe his assessment could apply equally well to American higher education.
"Our national system is, 'Do you have a degree or not?'" says Martin Scaglione, president of workforce development for ACT, a nonprofit education and career services company best known for its college entrance exams. "That doesn't really measure if you have skills."

Which skills and how to get them?

Just what skills define a good programmer is a subject of much debate. Some computer science graduates enter the workforce with a thorough theoretical understanding of software development but little practical experience. Others spent their college years hammering away at C++, Java, SQL, or other specific tools, but failed to gain any working knowledge of the higher concepts. And either type of programmer might lack the communication and teamwork skills necessary to become a top-performing employee.

At Zoho, Vembu is experimenting with a unique, homegrown approach to recruitment. Instead of hiring degreed workers -- who might have already picked up bad habits -- he starts with high school graduates and molds them into programmers. Often his candidates have no previous computing experience. Many don't even speak English. But once they've completed a two-year intensive study course developed at Zoho, Vembu says, they're fully prepared to work at any IT services company.

While a program as ambitious as Zoho's might work in India, where poverty is endemic, it would be impractical in the United States. Still, many American employers are trying a similar, ground-up approach to developer education. Rather than hiring new programmers to staff software projects, they recruit internally, often tapping employees with little or no previous coding experience to transition into development roles. Although the learning curve can be steep, such internal hires have the advantage of domain expertise and knowledge of business processes and objectives -- skills that could take green programmers even longer to master.

Aiding this transition is the proliferation of highly abstracted application frameworks, business process modeling tools, and rapid application development environments, all of which reduce the need for hard coding expertise. In many cases, a U.S.-based product manager can work with stakeholders to gather application

requirements and establish goals, leaving most of the heavy lifting to outsourced developers. The ability to act as liaison between coders and business managers is more important than a hard computer science background.

So will a computer science degree soon become a dead end in the workforce? Unlikely, as long as innovation remains the driving force of the knowledge-based economy. Still, the evidence suggests that pure computer science will increasingly be the domain of academia and research, with advanced degrees becoming the norm. Those with four-year degrees will want to back up their education with management and business skills, not to mention keep looking over their shoulders -- there may be high school students already nipping at their heels

At the concluding year of my high school I was asked which major I would pursue at university. I knew it; it's something about that magical device I've been sticking to for years. Come on, it's an easy guess. Yes, it's the computer. But... Wait a minute, please. What's the difference between Computer Engineering and Computer Science? Like I said, it's about "Computer", but I found two strange words attached to it and I had to make a choice. I was also told that the "Science" suffix would mean studying more of the software part while the "Engineering" suffix would mean studying more of the hardware part. I chose the science; Computer Science that is.

Now many years after graduation and exposure to different activities in the name of my degree in Computer Science, I've always spent a great deal of time pondering with the philosophical question: who am I? Not who am I in the literal meaning of my character, but, specifically, who am I while performing these activities as a computer "scientist". If you are one of those who are called Computer "Scientists" by degree, try it out; take this question with you as you work in your field and make a deliberate

focus on your activities with this question in mind. Will you be able to concisely answer that question to yourself? An answer that will always be true? Since I called on you to try this out, I will not answer it for you; instead, I will leap you ahead on some combination of results. If you work with algorithms, mathematical models and the like to help sequence DNA or to help in fighting Coronavirus, for instances, the answer to the question would be: I am a scientist (computing's one). If you work with the developing this or that application to company X or bank Y to automate work processes via cloud or mobile apps, the answer to your question would be: I am an engineer (software engineer). If you work with manipulating computer data, processes, or operating machines, you may answer to yourself: I am a computer technician. And I can enumerate many other names. The issue gets even worse if what you do involves a mix of all of these - hey, strap-in right there and hold on your sanity - make sure you don't suffer Dissociative Identity Disorder.

I will leave the argument as is from the above; would leave it just as an eye-opening question for you to ponder with and establish your own share of argument about who you are. That is, to stop taking your computer-science-based career for granted and to start questioning your true identity in your career. It's not for the sake of fostering discovering that it turned out you are a garbage collector after you long perceived yourself as a respectable scientist. No! It's for the sake of living a true identity without being fooled. If you found that who you are in your career is truthfully a Computer Technician. Do you accept it? If yes; then, no shame, good for you. If not, well, then, it's great that you started to ask yourself so that you can change roads to reach the true identity you designated to yourself from your pursuit of this field.

Aside from this self-centric questioning, it follows to ask from this investigation: is Computer Science as an independent field of study dead? Well, if what we do after graduating from Computer Science

school is scattered over multiple different specialties that we sometimes fail to uniquely identify ourselves in the course of our career; then, the question begs an answer. Debating an answer (and I said 'debating' for there can be no straightforward one) would require a philosophy of science treatise at a stretch of length; however, I would draw a potential line of inquiry. Computer science first started as a Mathematical undertaking. A possibility of relieving humans from the slavery of manual calculations. A "Beautiful Idea" as Leibniz perceived it. Then, starting with Alan Turing, it became a field of study - a powerful tool that only specialised people can take its reins. And now, as computing invaded every aspect of our lives and programming languages got higher and higher in levels of abstraction, computer science has started to fade out in the background. With these contemporary characteristics of the nature of the field in its real practice, do we still need a standalone Computer Science curriculum in which students admit to study programming, data structures, and some hardware in isolation from these real-life computing invasions? I believe it's futile to have a standalone Computer Science department/curriculum nowadays. While this would make the identity issue frustratingly persist, it will extremely delay talented Computer Scientists entrance in markets of computing research and applied computing applications that could help humanity- because simply no one would be competently ready to work in these with his bare bachelor degree. What I am saying is that if computing became an integral part of medicine, an integral part of data analysis, an integral part of military, an integral part of business, the old field name and curriculum Computer Science became too generic and irrelevant to prepare scientists in these areas. With computing sitting in the background of virtually every human activity, it's due time to rule out Computer Science and start calling for Computational Medicine, Computational Business Administration, Computational Neurology...etc. These different lines of engagements need young specialised computing scientists equipped with the right skill set and knowledge rather than the

one-field-fits-all approach of Computer Science education being followed in many areas of the world nowadays. We, Computer Scientists, also deserve to pursue just the type of problems we are interested in solving - and to uniquely be able to identify ourselves in the course of our career.